Script shell, les coups spéciaux

Hadouken !

LES BOUCLES

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Welcome $i times"
done
```

```
#!/bin/bash
for i in {1..5}
do
   echo "Welcome $i times"
done
```

```
while true
do
    statement(s)
done
```

```
#!/bin/bash
# set counter 'c' to 1 and condition
# c is less than or equal to 5
for (( c=1; c<=5; c++ ))
do
   echo "Welcome $c times"
done
```

```
PKGS="php7-openssl-7.3.19-r0  php7-common-7.3.19-r0
for p in $PKGS
do
   echo "Installing $p package"
   sudo apk add "$p"
done
```

https://www.cyberciti.biz/faq/bash-for-loop/

Les conditions

```
#!/bin/bash
# Basic if statement

if [ $1 -gt 100 ]
then
   echo Hey that\'s a large number.
   pwd
fi
```

Operator	Description
! EXPRESSION	The EXPRESSION is false.
-n STRING	The length of STRING is greater than zero.
-z STRING	The lengh of STRING is zero (ie it is empty).
STRING1 = STRING2	STRING1 is equal to STRING2
STRING1 != STRING2	STRING1 is not equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is numerically equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is numerically greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is numerically less than INTEGER2
-d FILE	FILE exists and is a directory.
-e FILE	FILE exists.
-r FILE	FILE exists and the read permission is granted.
-s FILE	FILE exists and it's size is greater than zero (ie. it is not empty).
-w FILE	FILE exists and the write permission is granted.
-x FILE	FILE exists and the execute permission is granted.

Code de sortie des commandes

Chaque programme à la fin de son exécution renvoie un code de sortie

- 0 tout va bien
- 1 erreur
- > 1 problème, regarder le manuel pour savoir à quoi correspond l'erreur

Pour connaître le code de sortie, il faut utiliser la variable **$?**

From : https://www.cyberciti.biz/faq/linux-bash-exit-status-set-exit-statusin-bash/

Affecter les variables

- variables vs environment variables
- export
  - export var=”toto”
- affecter la sortie d’une commande à une variable
  - var=$(ls /tmp)
- interpolation :
  - MY_VAR="42FileChecker"
  - echo "The name of the script is $MY_VAR"
  - echo "script_name_$MY_VAR_rocks"
  - echo "script_name_${MY_VAR}_rocks"

https://github.com/jgigault/bash-tips-and-tricks/blob/master/pages/bash-syntax/variables.md

passer des arguments à un script shell

3. Positional and special parameters

A shell script or a shell function shall be invoked with a list of arguments. We call it the **positional parameters** which are similar to the pointer to strings 'argv' in a C program. A positional parameter is like a variable whose name is a number other than 0. Calling your script with the command `./my_script "arg1" "arg2"` will declare two positional parameters `1` and `2` with the assigned value "arg1" and "arg2":

```
echo "$1"    # displays the first argument --> "arg1"
echo "$2"    # displays the second argument --> "arg2"
```

https://github.com/jgigault/bash-tips-and-tricks/blob/master/pages/bash-syntax/variables.md

EXO

Créons un nouveau script :

- le script prend maintenant un paramètre qui permettra de choisir le répertoire que l'on veut lister
- Si l'utilisateur n'a pas saisi de paramètre, lui en demander un et continuer à le demander tant que rien n'a été saisi
- Vérifier que le paramètre passé est un dossier
- Si ce n'est pas le cas, sortir en erreur 1
- Si c'est le cas, lister le dossier de façon récursive